Team Orange (Delta V Innovation Database Team)

Database Requirement Specifications

Authors:

Jonathan Cano:

-Introduction

-Project Preview

Jacob Huff:

-Development and Target Environments

-System Model

John Meyers:

-User Interaction

-Functional Requirements

-Nonfunctional Requirements

-Feasibility

-Conclusion

Customer: Mike Flamm, President of Delta V Innovations Inc.

TABLE OF CONTENTS

Introduction

The goal this project is to work with another group in updating existing software specializing in Computer Aided Design for crash and crime scene recreation, storage of captured data, and sharing amongst users: data, analysis and simulation of events.

We will build upon the previous work done on the Delta V database and make Mike Flamm's application, which currently collects car accident info and uses it to analyze and simulate the event. The features to be implemented this semester include: Merging two tables containing vehicle specs, updating vehicle specs table with the 2018-2019 car specs, creating tables/columns or queries for the group working on the Desktop Application. At the time this is written the group working on the Desktop application hasn't finished the design. This will be updated with more detail when the design is given to us. The main constraint is maintaining the database current efficiency and performance.

This document is intended to provide a detailed overview of application and database. It will describe the development environments, the system model, user interaction, functional and nonfunctional requirements, and the feasibility of the project.

Project Overview

The problem we are tasked with is merging two tables to reduce the number of searches for users, update the vehicle information with the 2018-2019 specs, and making changes to allow the desktop application being created by another group to function correctly. We will be developing scripts and methods to merge the tables and update them as accurately as possible.

Mike Flamm intends the users of the application to include insurance agencies, private investigators, and car manufacturers. A general overview of the application is that it takes in traffic conditions, time of day, car specs, etc... With that info the application seeks to help its users determine why the accident occurred and possibly run simulations on the event. The database of this project allows the user to store accident information as well as query information about vehicles. The previous database teams populated the database and created a communication protocol between the mobile app and the database.

The current problem with the database is that to query information on vehicle specs you must do a search on a vehicle specs table and then do another search on the vehicle_spec_additional table if you need more measurements for that vehicle. Merging these two table into one would solve this problem, but the info in the tables was acquired from different sources. This results in one table having a different format from the other, as well as the vehicle specs table having 40,000 more entries than the vehicle_spec_additional table. There is also no guarantee that every column in either table is filled in with information or that every row in the vehicle_spec_additional table will match with one in the vehicle spec, which forces us to develop different ways to merge these tables. Our solution for this problem is to extract information from certain columns in vehicle_spec_additional, which we can use to merge with the other table and then insert any rows that didn't merge successfully into the vehicle_specs table.

We are also tasked with general maintenance of the database, importing new vehicle information, and making necessary changes to the database to allow the desktop application being created by the other group to function correctly. This will be updated with more info when the design of the desktop application has been received.

The constraints for this project are the efficiency and storage of the database. We don't want to slow down the time it takes to retrieve or send information. We also don't want to make the database any bigger than necessary to maintain efficiency. There is also the issue of the data in the tables being missing or replicated.

Development and Target Environments

There are a couple of hardware and software resources that will be necessary to build and run the system. These resources currently include AWS and SQL. The system must also be able to communicate with both the desktop application as well as the mobile application in order to receive data from the mobile app and share that data with the desktop app or other users. The only resource needed by the end user in order to run the application will be a device to access it with.

The database is hosted on Amazon Web Services, or AWS. According to previous team's documentation, AWS was chosen over Microsoft Azure due to its ease of use (numerous online reference materials and tutorials) and added security features. AWS will only be used to host the database and to view certain statistics and information about the database. The actual data manipulation will be done using a tool called Heidi SQL. Heidi SQL is the tool we will be using throughout the semester to view the data in the database as well as perform the data manipulation that is needed by the customer. This will be done using SQL by writing queries to select, insert, add, and delete data.

End users will be able to retrieve or add data to the database using either an Android or iOS device. The app that will be used is available for free download on both types of devices. Using this application, users will be able to search for data using filters. The database will then

query based on the given filters and return the data. Users will also be able to add data to the database using the mobile application. This will allow for easy testing, as we can install the app and perform searches and insertions to and from the database as if we were the end user.

System Model

Previous teams have made great progress working on the database. Our team must first look at what previous teams have accomplished in order to get a better understanding of what we are going to be working with. Currently, the database contains large amounts of car information contained in two separate tables. This tables are well organized and contain information ranging from the early 1930s to the present day. End users also already have the ability to add or search for this data through the mobile application.

Our job will be to merge the two tables in the database that contain car information, accounting for duplicate and overlapping data between them. Currently, the end user must make two separate queries in order to get certain information due to it being in separate tables. Once the data is merged into a single, well organized table, the end user will be able to make a single query to retrieve the information that is needed. We will also add columns into the table based on information that the desktop team collects with its crash simulation application. If time permits, we will implement a password retrieval system as well as add new data from the 2019-2020 car information that comes out this year.

The system model and database diagrams can be seen in the appendix. We will be using the same model as previous teams throughout the semester.

User Interaction

The database has already been built and brought up to functionality. Our team will not be affecting how the database functions, merely how the data is stored within it. Two tables, both containing vehicle data, will be combined by our team. This will allow users to streamline their searching process by aggregating all their desired results into a single search, rather than two. Additionally, our team will remove duplicate entries in the tables and create a search function that returns the most conservative entries when multiple entries are valid search options. This will further streamline the user search process by removing the need for users to sift through multiple search results.

Functional Requirements

There are four main functional requirements we plan to accomplish this semester. These include splitting table columns to match existing table specifications, removing duplicate rows from existing tables, merging the two primary tables being used for vehicle information storage, and improving upon the existing vehicle data search function. Our current and primary goal is to merge the two tables containing vehicle information. This will allow for a singular search function to be implemented that will be effective across all user-relevant parts of the database. This merge is intended to add new information to entries in the existing table by merging it with a Canadian vehicle table with longer entries. In the case where a vehicle entry in the Canadian table is not already present in the existing database, new entries will be created to house that information. This will include rows that contain make, model, year, and no additional information. These entries are not useful to the user, and will likely not even appear in searches, but will improve the integrity of the database and facilitate future data additions. In order to accomplish this merge, the new vehicle information table must be reformatted to match the

structure of the existing table. This process will primarily consist of splitting a column in the new table that contains information held within multiple columns in the existing table. This split will result in two additional columns in the Canadian table – a model column and a trim column. The split will be such that the first part of the existing column will become the model column and the remainder will be put into trim. This should match the formatting of the model and trim columns in the existing table while ensuring that each entry in the model column is a real-world car model. Additionally, we will focus on removing duplicate and near-duplicate entries from both tables prior to the merge taking place. This step should be taken before the merge, as it will expedite the process of merging the tables. Deletion of entries will also necessitate communication with Mike in order to ensure that near-duplicate entries do not contain relevant information that should be retained. These steps will be made somewhat easier by using existing python code used by the previous database team that was used to add tables and edit table information within the database. Our final requirement is to improve upon the existing search function, primarily to streamline the user search procedure. Accomplishing this requirement will result in user searches no longer containing duplicate results or otherwise redundant information. This effectiveness and requirements for completing this search function will be dependent on the success of the merge. Additional information regarding the search will be added after the completion of the merge.

Nonfunctional Requirements

With the previous existence of the database we are working with and the data we are working with already having been entered into tables, many of the nonfunctional requirements for the tables we are working with have already been satisfied. In our update of the search function, we will need to take into account a few nonfunctional requirements.

While merging the data across the multiple tables holding vehicle data, we will need to ensure that there is consistency with how the data is stored within the tables. Another requirement is to ensure that the data does not exceed the storage capacity of the database. Since we are not adding additional data to the database, this requirement should not be an issue. The updated search function will have a few nonfunctional requirements we must take into account. Firstly, the search must occur within a reasonable time frame, ideally within seconds. Additionally, the search should output relevant, non-duplicate information. These requirements will be accomplished in part by ensuring the proper structure and integrity of the database are met prior to work on the search tool. There is a current level of security and encryption used while accessing the database. Our search function must operate within this limitation and not create any potential opportunity for a security risk.

Feasibility

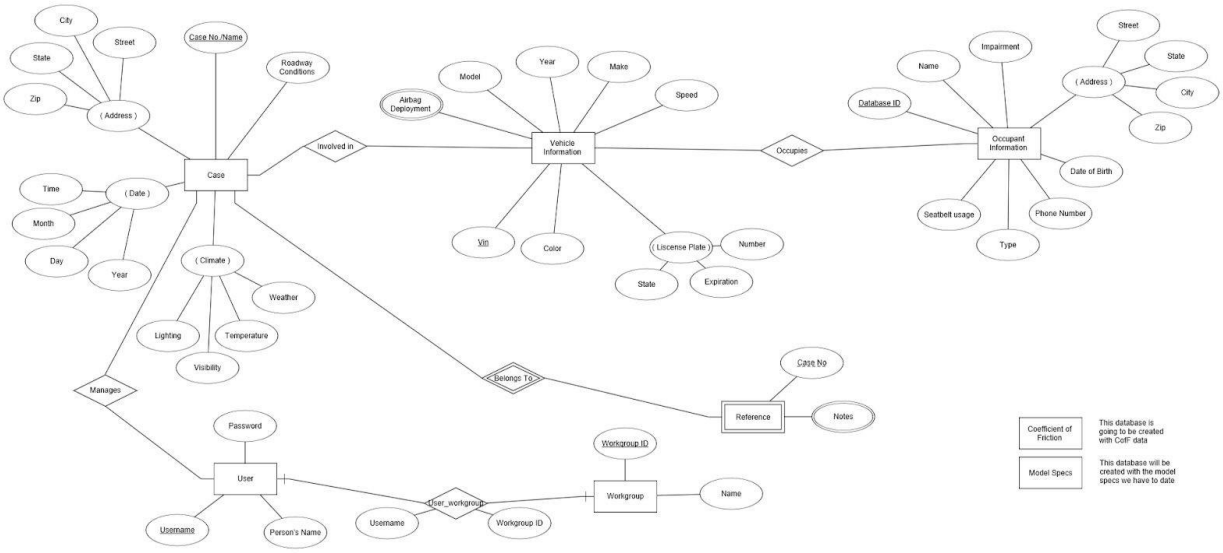
This project appears to be well within our team's realm of possibility. Our current estimate for completing the data merge is within three weeks, leaving the remainder of the project time for working on improving the search function. Our teams ability to meet weekly, along with Mikes quick responses to any questions or clarifications we may have, suggest to us that we should encounter few significant roadblocks to our timely completion of the project. We expect that even in the worst case scenario, we will have completed the merge of the two vehicle data tables. Under ideal circumstances, we will complete all the listed requirements and continue to find improvements that can be made to the search function.

Conclusion

The requirement specification document is now concluded. This specification will be referenced throughout the project and will be used as a guide when merging the tables in the database and when developing the improved search function.

Appendix

The following diagram is an Entity-Relationship model created by the database support team last year. We will be referencing this diagram as well as the relationship schema throughout the semester when creating new tables and editing the database.



Relational Schema

CASE(Case_No(PK), Address, Date, Climate, Roadway_Conditions, Lead_Investigator, Group)

VEHICLE(VIN(PK), License_No, Make, Model, Year, Speed, Airbag_Deploy, Color)

OCCUPANT_INFO(ID (PK), Name, Seatbelt, Impairment, Address, DOB, Phone_No, Type)

REFERENCE(Case_ID(FK), Witness_Notes, Personal_Notes)

MODEL_SPECS(Make(PK), Model(PK), Start_Year(PK), End_Year(PK), Front_Tire, Rear_Tire, Doors, Body_Style, Drive_Wheels, Curb_Weight, Weight_Front, Weight_Rear, 12 Length, Width, Height, Wheelbase, FOH, Front_Track, Rear_Track, Front_Bumper_Ht, Rear_Bumper_Ht, Ground_To_Base_Windshield, Stability_Ratio, CGH, Stability_Track)

USERS(Username(PK), Password, Pname)

WORKGROUPS(ID (PK), Name)

USER_WORKGROUP(Username(FK), WGID (FK))

Jacob Huff, Jonathan Cano, John Meyers, Tyler Dewitt

CS499-001, Spring 2019

2/22/2019

Word Count:

Jacob Huff: 550

Jonathan Cano: 697

John Meyers: 947